DAC_Phase4 : Water Quality Analysis Project

The goal of the "Water Quality Analysis Project" in Phase 3, is to perform preprocessing and Exploratory Data Analysis by plotting graphs and getting insights.

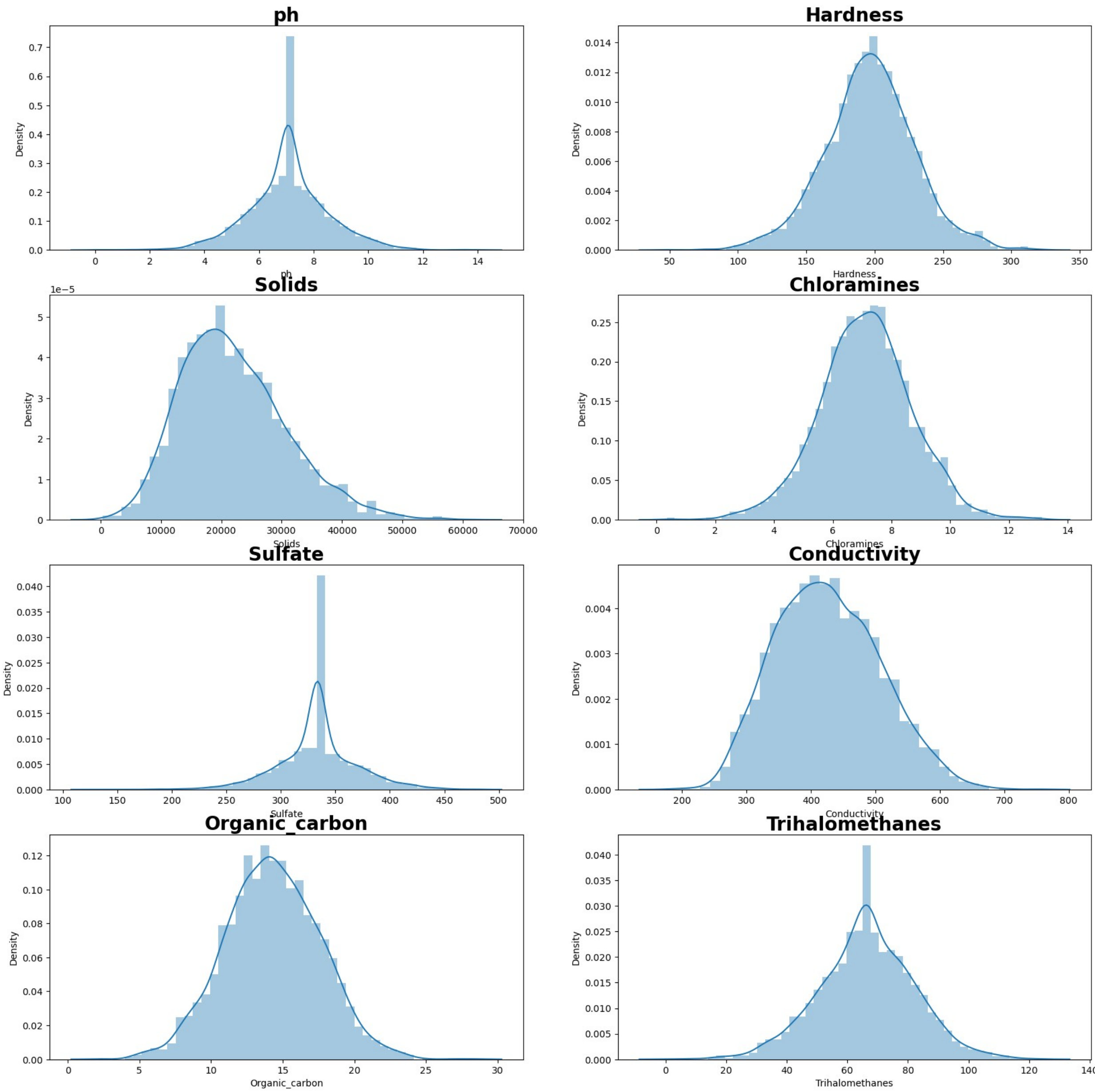
Our approach involves,

```
   1. finding correlation between the attributes of the dataset
provided,
   2. Handling missing values,
   3. Getting comparative insights by using necessary plots for
further processing and clear understanding on dataset attributes.
```

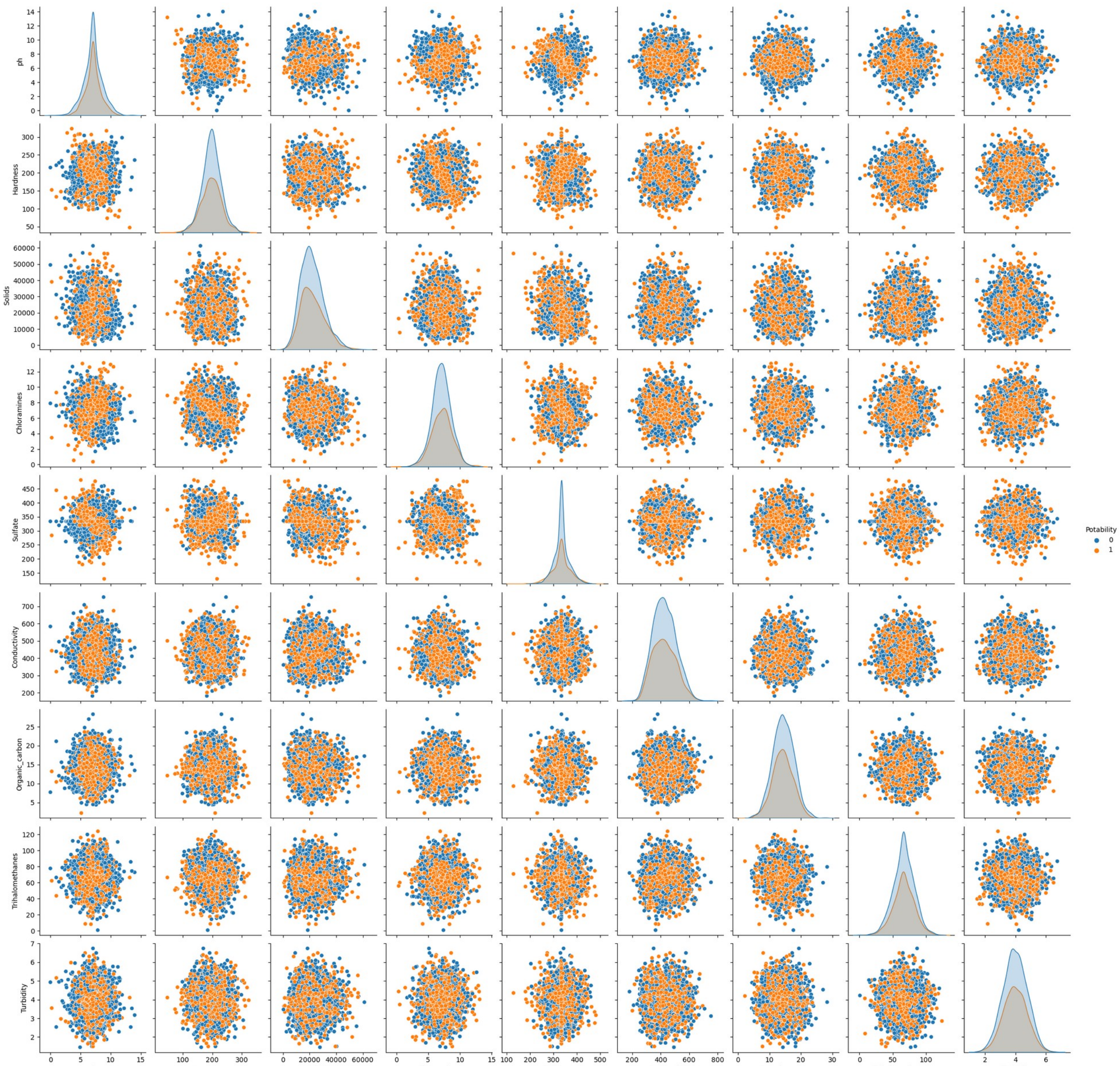
Phase_4

Feature Engineering

```
plt.figure(figsize=(20,20))
for i in range(8):
    plt.subplot(4,2,(i%8)+1)
    sns.distplot(df[df.columns[i]])

plt.title(df.columns[i],fontdict={'size':20,'weight':'bold'},pad=3)
plt.show()
```

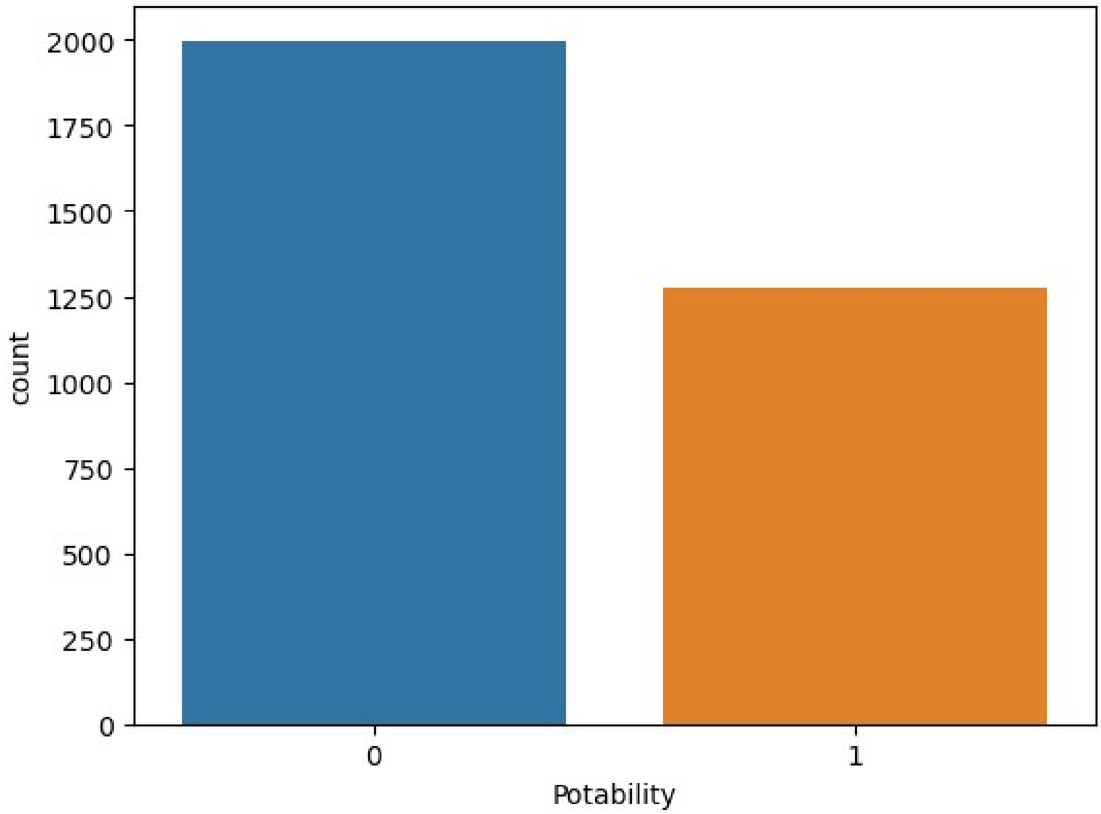



```
sns.pairplot(data=df, hue='Potability')

<seaborn.axisgrid.PairGrid at 0x1f75bf54b90>
```

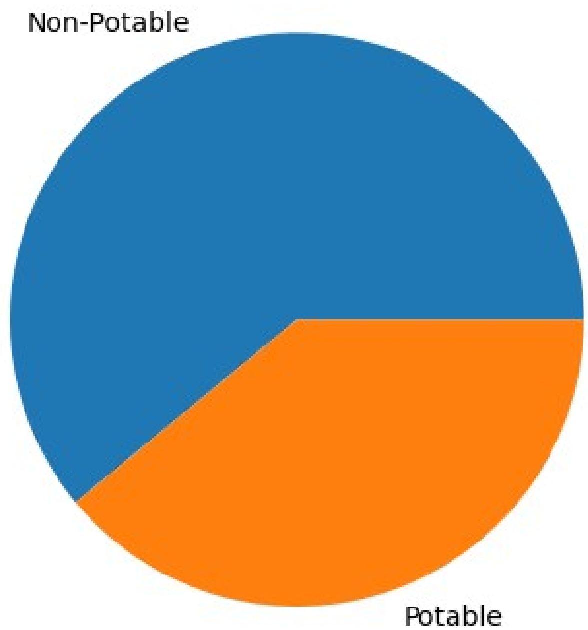



```
##Checking for distribution of Potable water

sns.countplot(x=df["Potability"])
```

```
<Axes: xlabel='Potability', ylabel='count'>
```




```
#Representing in a visually applealing pie chart

ratio = df.Potability.value_counts()
plt.pie(ratio, labels=['Non-Potable','Potable'])
plt.show()
```



MODEL Training and Evaluation

Seperating independent variable say X and dependent variable say Y

```
X =  df[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity']]
X.head()

         ph    Hardness        Solids  Chloramines     Sulfate
Conductivity  \
0  7.080795  204.890455  20791.318981     7.300212  368.516441
564.308654
1  3.716080  129.422921  18630.057858     6.635246  333.775777
592.885359
2  8.099124  224.236259  19909.541732     9.275884  333.775777
418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136
363.266516
4  9.092223  181.101509  17978.986339     6.546600  310.135738
398.410813

   Organic_carbon  Trihalomethanes  Turbidity
0       10.379783        86.990970   2.963135
1       15.180013        56.329076   4.500656
```

```
2       16.868637       66.420093   3.055934
3       18.436524      100.341674   4.628771
4       11.558279       31.997993   4.075075
```

```
y = df['Potability']
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Potability, dtype: int64
```

Splitting the dataset into Train and Test for modeling

```
from sklearn.model_selection import train_test_split
#splitting the dataset

X_train,X_test,Y_train,Y_test =
train_test_split(X,y,test_size=.2,random_state=42)
```

Importing necessary libraries for modeling and Evaluating

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

log_reg = LogisticRegression()

dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5)
```

Logistic Regression

```
log_reg.fit(X_train,Y_train)
tst2 = log_reg.predict(X_test)
```

Model accuracy

```
log_acc=accuracy_score(Y_test,tst2)

print("Train Set
```

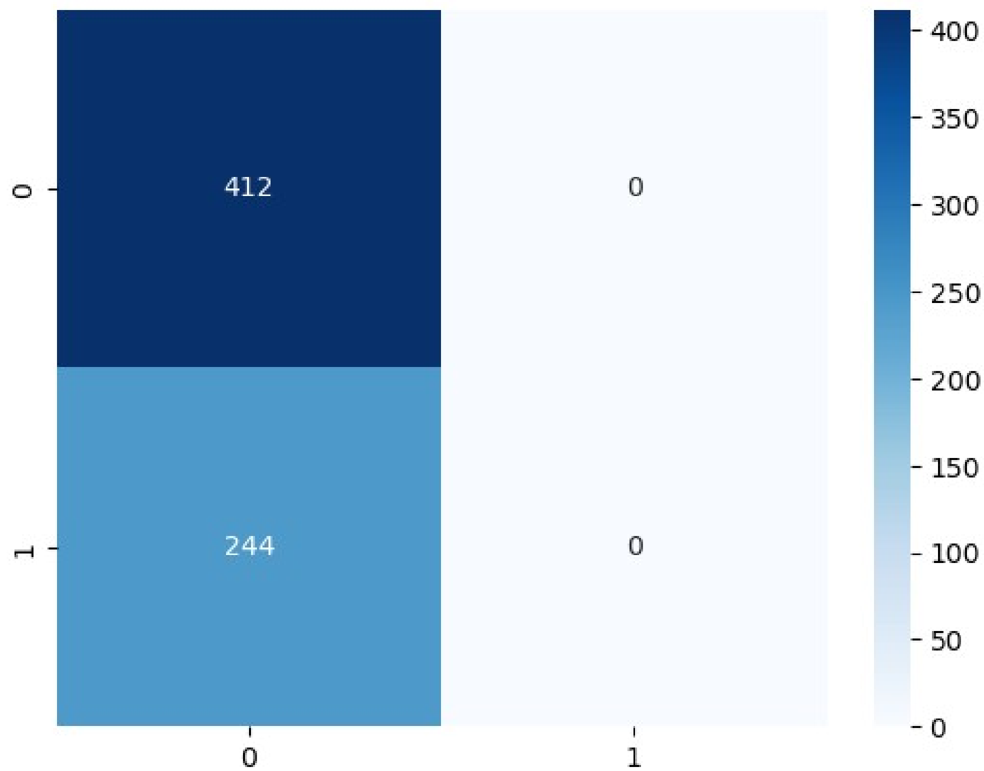
```
Accuracy:"+str(accuracy_score(Y_train,log_reg.predict(X_train))*100))
print("Test Set
Accuracy:"+str(accuracy_score(Y_test,log_reg.predict(X_test))*100))
```

```
Train Set Accuracy:60.57251908396947
Test Set Accuracy:62.80487804878049
```

Model Eevaluating

```
print('Logistic Regression\n')

log_cm = confusion_matrix(Y_test, tst2)
print(metrics.classification_report(Y_test, tst2))
sns.heatmap(log_cm, annot = True, fmt='d', cmap = 'Blues')
```

```
Logistic Regression

              precision    recall  f1-score   support

           0       0.63      1.00      0.77       412
           1       0.00      0.00      0.00       244

    accuracy                           0.63       656
   macro avg       0.31      0.50      0.39       656
weighted avg       0.39      0.63      0.48       656


<Axes: >
```



Decision Tree Classifier

```
dtc.fit(X_train, Y_train)
tst = dtc.predict(X_test)
```

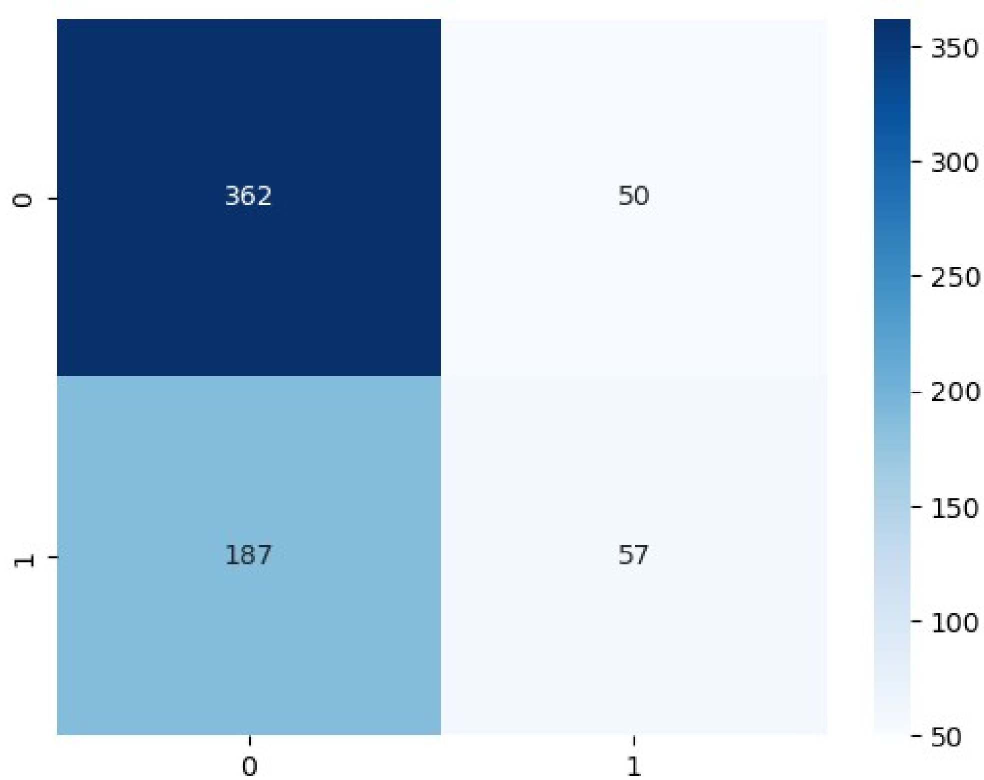
Model accuracy

```
dtc_acc= accuracy_score(Y_test,tst)

print("Train Set
Accuracy:"+str(accuracy_score(Y_train,dtc.predict(X_train))*100))
print("Test Set
Accuracy:"+str(accuracy_score(Y_test,dtc.predict(X_test))*100))

Train Set Accuracy:67.29007633587786
Test Set Accuracy:63.87195121951219
```

Model Eevaluating

```
print('Decision Tree\n')

decision_tree_cm = confusion_matrix(Y_test, tst)
print(metrics.classification_report(Y_test, tst))
sns.heatmap(decision_tree_cm, annot = True, fmt='d', cmap = 'Blues')
```

```
Decision Tree

              precision    recall  f1-score   support

           0       0.66      0.88      0.75       412
           1       0.53      0.23      0.32       244

    accuracy                           0.64       656
   macro avg       0.60      0.56      0.54       656
weighted avg       0.61      0.64      0.59       656


<Axes: >
```



Executing Feature Engineering

Try removing columns with many outliers

```
## We found ph, chloramine,solids columns having many outliers... And thus we train model without those data

X =  df[[ 'Hardness','Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity']]

X.head()
```

```
     Hardness     Sulfate  Conductivity  Organic_carbon
Trihalomethanes  \
0  204.890455  368.516441    564.308654       10.379783
86.990970
1  129.422921  333.775777    592.885359       15.180013
56.329076
2  224.236259  333.775777    418.606213       16.868637
66.420093
3  214.373394  356.886136    363.266516       18.436524
100.341674
4  181.101509  310.135738    398.410813       11.558279
31.997993

   Turbidity
0   2.963135
1   4.500656
2   3.055934
3   4.628771
4   4.075075
```

```
log_reg.fit(X_train,Y_train)
log_acc=accuracy_score(Y_test,log_reg.predict(X_test))


print("Train Set
Accuracy:"+str(accuracy_score(Y_train,log_reg.predict(X_train))*100))
print("Test Set
Accuracy:"+str(accuracy_score(Y_test,log_reg.predict(X_test))*100))
```

```
Train Set Accuracy:60.57251908396947
Test Set Accuracy:62.80487804878049
```

```
dtc.fit(X_train, Y_train)


dtc_acc= accuracy_score(Y_test,dtc.predict(X_test))

print("Train Set
Accuracy:"+str(accuracy_score(Y_train,dtc.predict(X_train))*100))
print("Test Set
Accuracy:"+str(accuracy_score(Y_test,dtc.predict(X_test))*100))
```

```
Train Set Accuracy:67.29007633587786
Test Set Accuracy:63.87195121951219
```

It seems to be same as the previous modeling

Phase_4 Conclusion

>> The two models Trained were Logistic regression model and Decision tree model

>> Out of the two models trained, *Decision Tree model* out performed Logistic Regression.

>> When we tried to improve the models by removing some columns which found to have many outliers and training the model again. this turned out the model to perform at same level.

>> From this move we can conclude that, from the given dataset, all the features or columns have same impact on the predictor variable and removing one thus slightly reduces the models performance.